Theoretical Computer Science 73 (1990) 319-328
North-Holland

NOTE

STRING MATCHING WITH WEIGHTED ERRORS

Alan A. BERTOSSI and Fabrizio LUCCIO
Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy

Elena LODI and Linda PAGLI
Dipartimento di Informatica e Applicazioni, Università di Salerno, 84081 Baronissi, Italy

Communicated by G. Ausiello
Received March 1989
Revised June 1989

Abstract. In the approximate string matching problem, differences are allowed between the pattern string P and each of its occurrences in the text string T, and one is interested in finding all the occurrences of P in T with at most k differences. We consider here weighted differences (errors) between P and T and develop fast sequential and parallel algorithms. In particular, we allow the following types of errors: mismatch whose weight depends on the mismatching characters, extra character with constant weight, missing character with constant weight, and transposition of two consecutive characters with constant weight. A set of theoretical results allows to extend known algorithms to solve this problem with $O(kn)$ sequential time and $O(k+\log m)$ parallel time on a PRAM model with $\max\{n+k+1, m^2\}$ processors, where k is the maximum sum of the error weights, n is the length of T, and m is the length of P.

1. Introduction

Let Σ be a finite alphabet, and let $P = p_1 \ldots p_m$ and $T = t_1 \ldots t_n$ be strings over Σ, $m \leq n$. The *approximate string matching* problem consists in finding all the occurrences of the *pattern* P in the *text* T, where, for a fixed integer k, up to k differences are allowed between P and each of its occurrences in T [13]. This problem is an important extension of exact string matching [7, 5, 1], and is theoretically relevant whenever T and/or P contain errors which may cause differences of the following types (see e.g. the survey [4]):

- d1 *mismatch* between a character p_i in P and the corresponding character t_j in T;
- d2 an *extra character* t_j in T, corresponding to no character in P;
- d3 an *extra character* p_i in P, corresponding to no character in T;
- d4 *transposition* of two adjacent characters $t_{j-1}t_j$ in T, which correspond to $p_{i-1}p_i$ in P, with $t_{j-1} \neq t_j$, $t_{j-1} = p_i$ and $t_j = p_{i-1}$.

Approximate string matching with differences d1, d2 and d3 has been considered in [15, 9, 4] and quickly solved in $O(kn)$ sequential time [8]. Weighted differences have also been considered in applications of pattern and speech recognition (see e.g. [12, 3]) but no fast sequential algorithms have been developed in this case. A general extension to context-dependent errors has been introduced in [2].

The *edit-distance* problem, namely, determining the minimum number of differences between two strings, is a recurrent subproblem of approximate string matching. Some algorithms have been devised to solve this problem, with weighted differences obeying specific assumptions (e.g. [11, 14, 16]). However, these algorithms do not help in the construction of efficient algorithms for string matching.

In this paper, we consider the approximate string matching problem with *weighted* errors e1, e2, e3 and e4 that are extensions of d1 to d4. These errors are defined as follows:

- e1 *mismatch*: as d1, with a weight function $w_1 : \Sigma^2 \to \mathbb{N}$ such that $w_1(p_i, t_j) = 0$ if and only if $p_i = t_j$;
- e2 *extra character in T*: as d2, with integer constant weight $w_2 > 0$;
- e3 *extra character in P*: as d3, with integer constant weight $w_3 > 0$;
- e4 *transition*: as d4, with integer constant weight $w_4 > 0$.

In the following we shall assume that, for any pair p_i, t_j, the function w_1 can be computed in constant time. Note that the set of errors considered here has not only a theoretical meaning, but also a practical interest. For example, the significance of error e1 becomes apparent if we assume that the text T has been typed through a keyboard whose keys correspond to the characters of Σ. A straightforward definition of w_1 can be given by the positions of any two keys on the board. Clearly, the probability of mismatching is higher between closer keys, that is, w_1 can be reasonably related to the distance between the keys. The constant weights of e2, e3 and e4, instead, derive from the assumption that, in first approximation, such errors are independent of the addressed keys, although the probabilities of the three errors are different.

In this paper we direct our attention to the following problem and give efficient algorithms for its solution.

Problem. Find all the occurrences of a pattern P in a text T under errors e1, e2, e3 and e4, such that the sum of the weights of the errors of each occurrence does not exceed a given integer $k \geq 0$.

As usual, finding the occurrences of P in T with overall error weight k will actually attain to finding all the positions of such occurrences, without reporting the specific errors incurred.

A nontrivial analysis will show that, with a proper extension of the algorithm given in [8], this problem can be solved in $O(kn)$ time and $O(wn)$ space, where $w = \max\{w_2, w_3, w_4\}$. Clearly, this $O(kn)$ time algorithm outperforms a straightforward $O(mn)$ time algorithm only when k is upper bounded by m. We also show

how our problem can be efficiently solved in parallel, in $O(k+\log m)$ time on a PRAM model with $\max\{n+k+1, m^2\}$ processors.

2. Basic techniques

To solve approximate string matching problems, it is useful to define an $(m+1)\times(n+1)$ matrix D as follows.

Definition 2.1. For all i, j, $1 \leq i \leq m$, $1 \leq j \leq n$, $D_{i,j}$ is the minimum value of the error weight between $p_1 \ldots p_i$ and any consecutive substring of the text T ending at t_j.

Matrix D is bordered with appropriate rows and columns, corresponding to initial conditions. Dynamic programming algorithms, designed in previous papers, are based on the property that $D_{i,j}$ only depends on $D_{i',j'}$ with $0 \leq i' \leq i$ and $0 \leq j' \leq j$, $D_{i,j}$ excluded. This is true also in our problem, where $D_{i,j}$ can be computed as:

$$
\begin{aligned}
&D_{0,j} := 0, \qquad 0 \leq j \leq n,\\
&D_{i,0} := i \cdot w_3, \quad 0 \leq i \leq m,\\
&D_{i,j} := \min\{D_{i-1,j-1} + w_1(p_i, t_j),\ D_{i,j-1} + w_2,\ D_{i-1,j} + w_3,\\
&\qquad \text{if } p_{i-1} \neq p_i \text{ and } p_{i-1} = t_j \text{ and } p_i = t_{j-1} \text{ then } D_{i-2,j-2} + w_4 \text{ else } \infty\},\\
&\qquad 1 \leq i \leq m, \quad 1 \leq j \leq n. \qquad (1)
\end{aligned}
$$

The possible dependencies between different values of D can be denoted by sequences of arrows between elements of D, called *paths*, in which $D_{i',j'} \rightarrow D_{i,j}$ indicates that $D_{i,j}$ is obtained from $D_{i',j'}$ in the minimization step. A *minimization path* is thus one path followed by the algorithm.

Straightforward methods for computing matrix D take $O(mn)$ time. Whenever entries on the same diagonals of D form nondecreasing sequences which increase in unit steps, faster algorithms running in $O(kn)$ time may be derived. In fact these algorithms actually compute at most k different values for each diagonal, namely, those corresponding to entries which produce an increment. This is the case of approximate string matching with differences d1, d2 and d3 [14, 8]. Since the values along the diagonals remain constant in the presence of matchings of subsequences of P and T, the algorithm must include an efficient way of finding the extremes of such subsequences without scanning them. In particular, the algorithm given in [8] actually computes another matrix, L, whose rows correspond to the diagonals of D and whose columns correspond to the number of differences in such diagonals. This matrix is evaluated in $O(kn)$ time by using a data structure called "suffix-tree" which allows a matching of maximal length between two substrings to be computed in $O(1)$ time [10, 6, 4]. In the next section we shall see that similar concepts can be applied to the efficient solution of our problem, by properly modifying the algorithms given in [8].

As a preliminary result we show that the values along the diagonals of D are nondecreasing even in the presence of weighted errors e1, e2 and e3 (not e4).

Theorem 2.2. *In the presence of errors e1, e2, e3, we have* $D_{i,j} - D_{i-1,j-1} \geq 0$, *for any pair i, j.*

Proof. The theorem is trivially true when $i = 1$ and $1 \leq j \leq n$, since $D_{0,j-1} = 0$ and the error weights are nonnegative. Therefore, assume that $i \geq 2$ and let us proceed by induction on $i+j$.

$D_{i,j}$ is obtained in one of the following ways:

(i) $D_{i,j} = D_{i-1,j-1} + w_1(x_i, t_j)$. The theorem trivially holds.

(ii) $D_{i,j} = D_{i-1,j} + w_3$. By induction hypothesis, we have $D_{i-1,j} \geq D_{i-2,j-1}$. Hence, $D_{i,j} \geq D_{i-2,j-1} + w_3$. Since $D_{i-1,j-1} \leq D_{i-2,j-1} + w_3$, we obtain $D_{i,j} \geq D_{i-1,j-1} - w_3 + w_3 = D_{i-1,j-1}$, as required.

(iii) $D_{i,j} = D_{i,j-1} + w_2$. This case is symmetric to case (ii). Only observe that for $j = 1$ the theorem trivially holds since $D_{i,1} = D_{i,0} + w_2 = iw_3 + w_2 > (i-1)w_3 + w_2 = D_{i-1,0} + w_2 > D_{i-1,0}$. □

Theorem 2.2 is indeed an extension of a similar result mentioned in [16] for the edit distance problem. In fact, the proof follows the lines of Lemma 3 of the above paper.

Theorem 2.2 shows that approximate string matching with errors e1, e2, e3 can be solved in time $O(kn)$, by appropriately changing the algorithm given in [8]. In the present case we must use the dynamic program (1) where the "min" is computed without the last "if" line relative to transpositions. For this purpose, assign number d to the diagonal of D consisting of all the entries $D_{i,j}$ for which $j - i = d$. Then, define $L_{d,e}$ as follows.

Definition 2.3. For all pairs d, e, with $0 \leq e \leq k$, $-m \leq d \leq n$, $L_{d,e}$ is the maximum row index i of D such that $D_{i,j} \leq e$ and $j - i = d$.

As in [8], the computation of $L_{d,e}$ can be carried out column-by-column, that is, all the values $L_{d,e}$, for a given e, can be computed from those of $L_{d,e'}$, $e' < e$. Since $D_{i,j}$ may be larger than $D_{i-1,j-1}+1$, a given error value e may be not present in the diagonal d of D. In this case $L_{d,e}$ indicates the maximum row index for which $D_{i,j} = e' < e$ (see Definition 2.3). For this reason, we also need to maintain the exact value of the error weight for each $L_{d,e}$. We store such values into another matrix E defined as follows.

Definition 2.4. $E_{d,e}$ is the value of $D_{i,j}$ such that $i = L_{d,e}$ and $j - i = d$.

To understand the mechanism of computation of $L_{d,e}$ and $E_{d,e}$, let us interpret it on matrix D, as follows. Let i_1, i_2 and i_3 be the maximum row indices, assuming

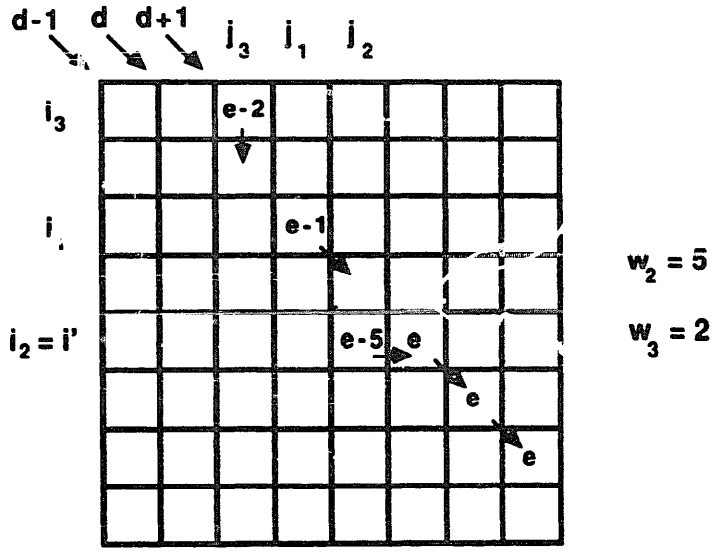

Fig. 1. The generation of the lowest value e on diagonal d of D.

that they all exist, such that $D_{i_1,j_1} = e-1$, $j_1 - i_1 = d$; $D_{i_2,j_2} = e - w_2$, $j_2 - i_2 = d-1$; $d_{i_3,j_3} = e - w_3$, $j_3 - i_3 = d+1$. Then, $i' = \max\{i_1+1, i_2, i_3+1\}$, if $D_{i_1,j_1} + w_1(p_{i_1+1}, t_{j_1+1}) = e$, or $i' = \max\{i_2, i_3+1\}$, otherwise, is a row index such that $D_{i',j'} = e$, $j' - i' = d$.

In the example of Fig. 1, we have $i' = i_2$, $j' = j_2 + 1$, that is a value e is generated in D_{i_2,j_2+1}. Once i' has been computed, we then proceed from $D_{i',j'}$ on diagonal d, testing for matchings between $p_{i'+h}$ and $t_{j'+h}$, $h = 0, 1, \ldots, q$, which maintain the error equal to e. We then have $L_{d,e} = E_{d,e} = D_{i'+q,j'+q}$. If i' does not exist, no element $D_{i,j}$ on diagonal d assumes the value e, hence $L_{d,e}$ takes the same value as $L_{d,e-1}$.

In the next section we will show that, if error e4 is also considered, the values along the diagonals of D may decrease. However, some properties of this matrix allow the problem to be solved in time $O(kn)$ in this case too.

3. Solving the general problem

Let us now consider our problem, where transposition errors e4 are added to the errors e1, e2, e3. Matrix D will be computed by the dynamic program (1), however the present situation is strongly complicated by the fact that consecutive values on the diagonals of D may decrease.

We preliminarly note that transposition will never be recognized as the error leading from $D_{i-2,j-2}$ to $D_{i,j}$, if its weight is higher than the sum of weights of any alternative path leading from $D_{i-2,j-2}$ to $D_{i,j}$. In fact, let $w_1^* = \max\{w_1(x, y): x \neq y\}$. If $w_4 \geq \min\{w_2 + w_2, 2w_1^*\}$ we have $D_{i-2,j-2} + w_4 > D_{i,j}$ for all i, j and only errors e1, e2, e3 will be detected. In this case the technique indicated in the previous section still applies, leading to a straightforward $O(kn)$ time solution to the present problem. To obtain the same time bound for $w_4 < \min\{w_2 + w_3, 2w_1^*\}$, we must prove some new results.

Theorem 3.1. *In the presence of errors e1, e2, e3, e4, we have, for any pair i, j:*

$$D_{i,j} - D_{i-1,j-1} \geq \begin{cases} 0 & \text{if } D_{i,j} \text{ is not obtained by transposition,} \\ w_4 - \min\{w_2 + w_3, w_1^*\} & \text{if } D_{i,j} \text{ is obtained by transposition.} \end{cases}$$

Proof. As in the proof of Theorem 2.2, we proceed by induction on $i+j$. We have four cases:

(a) $D_{i,j} = D_{i-1,j-1} + w_1(p_i, t_j)$. The theorem trivially holds.

(b) $D_{i,j} = D_{i-1,j} + w_3$. We have two subcases:

(b1) $D_{i-1,j}$ has not been obtained by transposition. Thus by inductive assumption we have $D_{i-2,j-1} \leq D_{i-1,j}$ and the inequality $D_{i-1,j-1} \leq D_{i,j}$ easily follows as in the proof of Theorem 2.2 (case ii).

(b2) $D_{i-1,j}$ has been obtained by transposition. That is, $D_{i-1,j} = D_{i-3,j-2} + w_4$. By assumption we also have that $D_{i-1,j-1} \leq D_{i-2,j-2}$, since $p_{i-1} = t_{j-1}$. Thus $D_{i-1,j-1} \leq D_{i-3,j-2} + w_3$. Now the inequality $D_{i-1,j-1} \leq D_{i,j}$ easily follows.

(c) $D_{i,j} = D_{i,j-1} + w_2$. The proof is analogous to case (b).

(d) $D_{i,j} = D_{i-2,j-2} + w_4$. We have four subcases.

(d1) $D_{i-1,j-1} = D_{i-2,j-1} + w_3$. We have $D_{i-1,j-1} \leq D_{i-2,j-2} + w_2 + w_3$; therefore, $D_{i,j} - D_{i-1,j-1} \geq D_{i-2,j-2} + w_4 - D_{i-2,j-2} - w_2 - w_3 = w_4 - (w_2 + w_3)$.

(d2) $D_{i-1,j-1} = D_{i-1,j-2} + w_2$. As in subcase (d1) we derive $D_{i,j} - D_{i-1,j-1} \geq w_4 - (w_2 + w_3)$.

(d3) $D_{i-1,j-1}$ has been obtained from $D_{i-2,j-2}$. We have $D_{i-1,j-1} \leq D_{i-2,j-2} + w_1^*$; hence: $D_{i,j} - D_{i-1,j-1} \geq D_{i-2,j-2} + w_4 - D_{i-2,j-2} - w_1^* = w_4 - w_1^*$.

Combining subcases (d1)-(d3) we have $D_{i,j} - D_{i-1,j-1} \geq w_4 - \min\{w_2 + w_3, w_1^*\}$, from which the theorem follows.

(d4) $D_{i-1,j-1}$ has been obtained by transposition from $D_{i-3,j-3}$. Therefore, $D_{i-1,j-1} = D_{i-3,j-3} + w_4$, hence $D_{i,j} - D_{i-1,j-1} = D_{i-2,j-2} + w_4 - D_{i-3,j-3} - w_4 = D_{i-2,j-2} - D_{i-3,j-3}$, and the theorem follows by induction hypothesis. □

Corollary 3.2. *In the presence of errors e1, e2, e3, e4, if $D_{i,j} < D_{i-1,j-1}$, then $D_{i+1,j+1} \geq D_{i,j}$.*

Proof. By contradiction, if we had $D_{i+1,j+1} < D_{i,j}$, then, by Theorem 3.1, $D_{i+1,j+1}$ should be derived from $D_{i-1,j-1}$ by transposition. However, in this case we would have $D_{i+1,j+1} = D_{i-1,j-1} + w_4$, hence $D_{i+1,j+1} > D_{i,j} + w_4$ by hypothesis, against the assumption. □

Corollary 3.2 shows that the diagonal values cannot have two consecutive decrements. The main property of such values is given in the next theorem, which shows that values followed by a decrement can be neglected in the computation of the successive values of D.

Theorem 3.3. *In the presence of errors e1, e2, e3, e4, if $D_{i,j} < D_{i-1,j-1}$, then each of the values $D_{i',j'}$, with $i' \geq i-1$, $j' \geq j-1$, excluding $D_{i-1,j-1}$, can be generated through a minimization path which does not include $D_{i-1,j-1}$.*

Proof. By Theorem 3.1, if $D_{i,j} < D_{i-1,j-1}$, there is a transposition $p_{i-1}p_i = t_j t_{j-1}$. There are four entries of D which could be generated from $D_{i-1,j-1}$, namely, $D_{i-1,j}$, $D_{i,j-1}$, $D_{i,j}$, and $D_{i+1,j+1}$.

(i) Consider $D_{i-1,j}$. Since $p_{i-1} = t_j$, then $D_{i-1,j} \leq D_{i-2,j-2} + w_2$. Hence $D_{i-1,j}$ cannot be derived from $D_{i-1,j-1}$.

(ii) Consider $D_{i,j-1}$. Since $p_i = t_{j-1}$, then $D_{i,j-1} \leq D_{i-2,j-2} + w_3$. Thus $D_{i,j-1}$ cannot be derived from $D_{i-1,j-1}$.

(iii) By assumption, $D_{i-1,j-1} > D_{i,j}$. Therefore, $D_{i,j}$ cannot be generated by $D_{i-1,j-1}$.

(iv) Finally, consider $D_{i+1,j+1}$. In this case, there is also a transposition $p_i p_{i+1} = t_{j+1}t_j$, hence $p_{i+1} = p_{i-1}$ and $t_{j+1} = t_{j-1}$. Four possibilities may come up, depending on the entry from which $D_{i-1,j-1}$ is derived.

(a) $D_{i-1,j-1}$ is derived from $D_{i-2,j-1}$. Then $D_{i-1,j-1} = D_{i-2,j-1} + w_3$. Since $p_{i-1} = t_j$ and $p_i = t_{j+1}$, then $D_{i,j+1} \leq D_{i-2,j-1}$. Thus $D_{i+1,j+1} \leq D_{i,j+1} + w_3 \leq D_{i-2,j-1} + w_3 = D_{i-1,j-1}$. Therefore, $D_{i+1,j+1}$ cannot be derived from $D_{i-1,j-1}$ by a transposition.

(b) $D_{i-1,j-1}$ comes from $D_{i-1,j-2}$. Then $D_{i-1,j-1} = D_{i-1,j-2} + w_2$ Since $p_i = t_{j-1}$ and $p_{i+1} = t_j$, then $D_{i+1,j} \leq D_{i-1,j-2}$. Thus $D_{i+1,j+1} = D_{i+1,j} + w_2 \leq D_{i-1,j-2} + w_2 = D_{i-1,j-1}$. Again, $D_{i+1,j+1}$ cannot be derived from $D_{i-1,j-1}$ by a transposition.

(c) $D_{i-1,j-1}$ is derived from $D_{i-2,j-2}$. Then $D_{i-1,j-1} = D_{i-1,j-2} + w_1(p_{i-1}, t_{j-1})$. Since $p_{i-1} = p_{i+1}$ and $t_{j-1} = t_{j+1}$, then we must have $w_1(p_{i-1}, t_{j-1}) = w_1(p_{i+1}, t_{j+1})$. $D_{i+1,j+1}$ can be obtained as $D_{i,j} + w_1(p_{i+1}, t_{j+1}) \leq D_{i-2,j-2} + w_4 + w_1(p_{i+1}, t_{j+1})$. This value of $D_{i+1,j+1}$ is not greater than that obtained from $D_{i-1,j-1}$ by a transposition, hence such a transposition can be disregarded.

(d) $D_{i-1,j-1}$ comes from $D_{i-3,j-3}$, hence $D_{i-1,j-1} = D_{i-3,j-3} + w_4$. We have $D_{i,j} \leq D_{i-1,j-1}$ by hypothesis, and $D_{i,j} = D_{i-2,j-2} + w_4$ by Theorem 3.1, that is $D_{i-3,j-3} > D_{i-2,j-2}$, hence also $D_{i-2,j-2}$ comes from a transposition. In general, whenever $D_{x,y} > D_{x+1,y+1}$ and $D_{x,y}$ derives from $D_{x-2,y-2}$ by a transposition, also $D_{x-1,y-1}$ must derive from $D_{x-3,y-3}$ by a transposition. Therefore let us consider the longest sequences of transpositions such that $D_{i-h,j-h} \to D_{i-h+2,j-h+2} \to \cdots \to D_{i-4,j-4} \to D_{i-2,j-2}$, and $D_{i-h+1,j-h+1} \to D_{i-h+3,j-h+3} \to \cdots \to D_{i-3,j-3} \to D_{i-1,j-1}$. Obviously, both $D_{i-h,j-h}$ and $D_{i-h+1,j-h+1}$ are not obtained by transposition and $D_{i-h,j-h} < D_{i-h+1,j-h+1}$. By points (a), (b) and (c), $D_{i-h+3,j-h+3}$ can be obtained via a path which does not include $D_{i-h+1,j-h+1}$. This reasoning can be iterated to all entries in the sequence $D_{i-h+3,j-h+3}$, $D_{i-h+5,j-h+5}, \ldots, D_{i-1,j-1}$, $D_{i+1,j+1}$. Therefore, $D_{i+1,j+1}$, can be derived via a path not including $D_{i-1,j-1}$. □

Based on Theorem 3.3, we can solve our string matching problem by extending the algorithm given in [8]. The following algorithm SMWE (for "String Matching with Weighted Errors") computes matrices L and E by directly implementing the mechanism presented in Section 2, with the provision that diagonal values followed by decrements are never computed.

Algorithm SMWE applies to a text and a pattern delimited with special marks, namely $t_0 = t_{n+1} = \cdots = t_{n+m} = \$$, $p_0 = \$$, and uses an extended definition of the

function w_1 by letting $w_1(x, \$) = 0$, for all $x \in \Sigma$. The algorithm borders matrices L and E with row $n+1$ and columns $-w, -w+1, \ldots, -1$. Note that Step 3′ copies the last value determined on diagonal d, namely $L_{d,e-1}$ ($E_{d,e-1}$), in $L_{d,e}$ ($E_{d,e}$, respectively) because diagonal d does not contain the value e of the error weight. Step 3″ tests whether a row index $\geq$ row has been already computed on diagonal d, for which the error weight value is less than e; in this case $L_{d,e-1}$ ($E_{d,e-1}$) is the correct value to be assigned to $L_{d,e}$ ($E_{d,e}$).

Although Algorithm SMWE is quite different from the one given in [8] (namely the core Steps 3, 3′, and 3″ are peculiar to SMWE), its correctness easily follows by a similar argument. Its complexity is also the same, since SMWE tests for matchings in Step 4, and in the inner loop of Step 2 in exactly the same way as in [8]. This fact, together with Theorems 2.2, 3.1 and 3.3, allows a complexity of $O(k \cdot n)$ to be attained. Indeed, it is again possible to use a suffix-tree data structure to implement Step 4 and thus find $L_{d,e}$ in O(1) time. Since e can assume at most k different values for each diagonal, the overall complexity is $O(kn)$. The space required by SMWE is $O(w \cdot n)$, with $w = \max\{w_2, w_3, w_4\}$, since columns $e-1, \ldots, e-w$ of L and E must be maintained to compute column e.

Algorithm SMWE

Step 1. $w := \max\{w_2, w_3, w_4\}$;
for $d := 0$ **to** n **do**
for $e := -w$ **to** -1 **do** $L_{d,e} := -1$; $E_{d,e} := -\infty$ **enddo**
enddo;
for $d := -(\lfloor k/w_3 \rfloor + 1)$ **to** -1 **do**
for $e := (|d|-1) \cdot w_3 - w_2$ **to** $|d| \cdot w_3 - 1$ **do** $L_{d,e} := -\infty$; $E_{d,e} := -\infty$ **enddo**
$L_{d,(|d| \cdot w_3 - 1)} := |d| - 1$; $E_{d,(|d| \cdot w_3 - 1)} := |d| \cdot w_3$
enddo;
for $e := -w$ **to** k **do** $L_{n+1,e} := -\infty$; $E_{n+1,e} := -\infty$ **enddo**;

Step 2. **for** $d := 0$ **to** n **do**
row $:= 0$; $E_{d,0} := 0$;
while $p_{\text{row}+1} = t_{\text{row}+d+1}$ **do** row $:=$ row $+ 1$ **enddo**;
$L_{d,0} :=$ row
enddo;

Step 3. **for** $e := 1$ **to** k **do**
for $d := -\lfloor e/w_3 \rfloor$ **to** n **do**
$r := L_{d,e-1}$; $s := L_{d,e-w_4}$;
$e_1 := E_{d,e-1} + w_1(p_{r+1}, t_{r+d+1})$; $r_1 := r+1$;
$e_2 := E_{d-1,e-w_2} + w_2$; $r_2 := L_{d-1,e-w_2}$;
$e_3 := E_{d+1,e-w_3} + w_3$; $r_3 := L_{d+1,e-w_3} + 1$;
if $p_s \neq p_{s+1}$ **and** $p_s = t_{s+d+1}$ **and** $p_{s+1} = t_{s+d}$
then $e_4 := E_{d,e-w_4} + w_4$; $r_4 := s+2$
else $e_4 := \infty$
endif;

Step 3'. $S := \{e_i : e_i = e\}$;
if $S = \emptyset$ **then**
$E_{d,e} := E_{d,e-1}$; $L_{d,e} := L_{d,e-1}$
else
row $:= \max\{r_i : e_i \in S\}$;
Step 3''. **if** $r \geq$ row **then**
$E_{d,e} := E_{d,e-1}$; $L_{d,e} := L_{d,e-1}$
else
Step 4. **while** $p_{\text{row}+1} = t_{\text{row}+d+1}$ **do** row $:=$ row $+ 1$ **enddo**
Step 5. $L_{d,e} =$ row; $E_{d,e} := e$
endif
endif
Step 6. **if** $L_{d,e} = m$ **and** $d \leq n - m$ **then**
print *THERE IS AN OCCURRENCE OF THE PATTERN IN THE TEXT ENDING AT t_{d+m} WITH ERROR WEIGHT $E_{d,e}$*
endif
enddo
enddo.

The string matching problem with errors e1, e2, e3 and e4 can also be solved by an extension of the parallel algorithm given in [8], under the computational model of CREW PRAM. In fact, one RAM processor per diagonal can be used to compute the values of $L_{d,e}$, which now depend on $L_{d,e-1}$, $L_{d-1,e-w_2}$, $L_{d+1,e-w_3}$ and $L_{d,e-w_4}$. Furthermore, the values of $E_{d,e}$ must also be computed by the same processor. As in [8], the algorithm requires $O(k + \log m)$ PRAM steps, using $O(n + m^2)$ processors, and is of interest when $k \leq c \log m$.

4. Concluding remarks

In this paper we have considered the introduction of a set of weighted errors in the problem of approximate string matching. On the grounds of some theoretical results on matrix D, we have been able to devise efficient sequential and parallel algorithms for the solution of this problem, as extensions of known algorithms for unweighted approximate string matching.

A variety of open problems remain. In fact, the set of errors considered in this paper is to be seen as an example, albeit important, of the many different situations that may occur in a text-editing or information retrieval environment. Other errors should be considered for similar applications, by extending the weights of errors e2, e3 and e4 to nonconstant functions. In this case, however, new techniques should be devised to build algorithms more efficient than the straightforward application of dynamic programming.

References

[1] A. Apostolico and R. Giancarlo, The Boyer-Moore-Galil string searching strategies revisted, *SIAM J. Comput.* **15** (1986) 98-105.

[2] A.A. Bertossi, E. Lodi, F. Luccio and L. Pagli, Approximate string matching with context-dependency, in: *Proc. 25th Ann. Allerton Conference*, Monticello, IL (1987) 46-53.

[3] F. Charot, P. Frison and P. Quinton, Systolic architectures for connected speech recognition, *IEEE Trans. Acoustic, Speech Signal Proc.* **34** (1986) 765-779.

[4] Z. Galil and R. Giancarlo, Data structures and algorithms for approximate string matching, *J. Complexity* **4** (1988) 33-72.

[5] Z. Galil and J. Seiferas, Time-space-optimal string matching, *J. Comput. System Sci.* **26** (1983) 280-294.

[6] D. Harel and R.E. Tarjan, Fast algorithms for finding nearest common ancestors, *SIAM J. Comput.* **13** (1984) 338-355.

[7] D.E. Knuth, J. H. Morris and V.R. Pratt, Fast pattern matching in strings, *SIAM J. Comput.* **6** (1977) 322-350.

[8] G.M. Landau and U. Vishkin, Introducing efficient parallelism into approximate string matching and a new serial algorithm, in: *Proc. 18th Ann. ACM Symp. on Theory of Computing*, Berkeley, CA (1986) 220-230.

[9] G.M. Landau and U. Vishkin, Fast string matching with k differences, *J. Comput. System. Sci.* **37** (1988) 63-78.

[10] E.M. McCreight, A space economical suffix tree construction algorithm, *J. ACM* **23** (1976) 262-272.

[11] W.J. Masek and M.S. Paterson, A faster algorithm computing string edit distances, *J. Comput. System Sci.* **20** (1980) 18-31.

[12] B.J. Oommen, Recognition of noisy subsequences using constrained edit distances, *IEEE Trans. Pattern Anal. Mach. Intel.* **9** (1987) 676-685.

[13] D. Sankoff and J.B. Kruskal, eds., *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison* (Addison Wesley, Reading, MA, 1983).

[14] E. Ukkonen, On approximate string matching, Lecture Notes in Computer Science **158** (Springer, Berlin, 1983) 487-495.

[15] E. Ukkonen, Finding approximate patterns in strings, *J. Algorithms* **6** (1985) 132-137.

[16] E. Ukkonen, Algorithms for approximate string matching, *Inform. and Control* **64** (1985) 100-118.